Programmieren 1

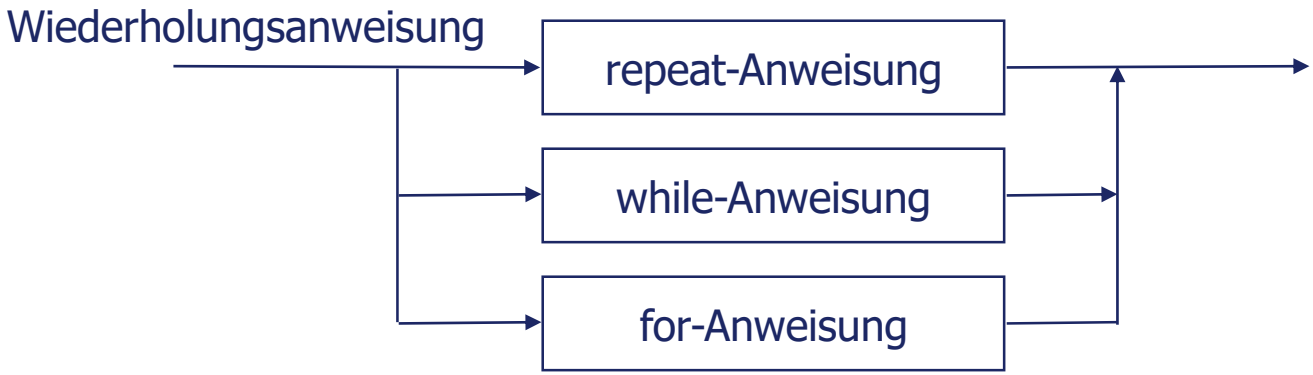
Wintersemester 2016/2017

Marcus Riemer, B.Sc.

Basierend auf den Unterlagen „Programmstrukturen 1“
von Prof. Dr. Andreas Häuslein

5 Wiederholungsanweisungen

- Dienen dazu, die algorithmische Struktur *Iteration* in Programmen umzusetzen
- Steuern als Kontrollstrukturen, **wie oft** eine Anweisung ausgeführt werden soll (Schleifen)
- In Pascal gibt es drei Arten von Wiederholungsanweisungen:



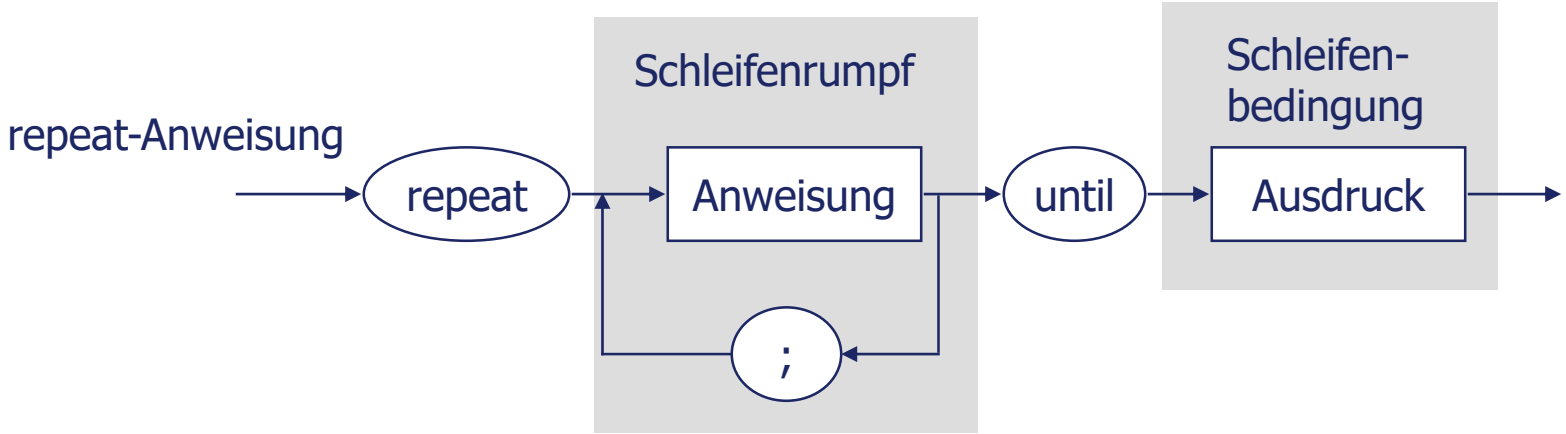
5 Wiederholungsanweisungen

- Genereller Aufbau
  - Zwei wesentliche Bestandteile:
    - *Schleifenbedingung*: steuert die Anzahl der Wiederholungen (Schleifendurchläufe)
    - *Schleifenrumpf*(oder -körper): beinhaltet die zu wiederholenden Anweisungen

- Genereller Ablauf
  - Schleifenbedingung wird vor/nach jedem Schleifendurchlauf ausgewertet
  - Sofern die Schleifenbedingung einen erneuten Schleifendurchlauf zulässt, werden *alle* Anweisungen des Schleifenrumpfes nochmals ausgeführt
  - Es erfolgt eine erneute Auswertung der Schleifenbedingung (siehe oben)
  - Prozeduren `Break` oder `Continue` dürfen nicht verwendet werden!

5 `repeat`-Anweisung

- Realisiert eine Iteration mit Abbruchbedingung
- Syntaktischer Aufbau:



Achtung, syntaktische Feinheiten!

- Mehrere Anweisungen im Schleifenrumpf
  OHNE Klammerung durch `begin` und `end`

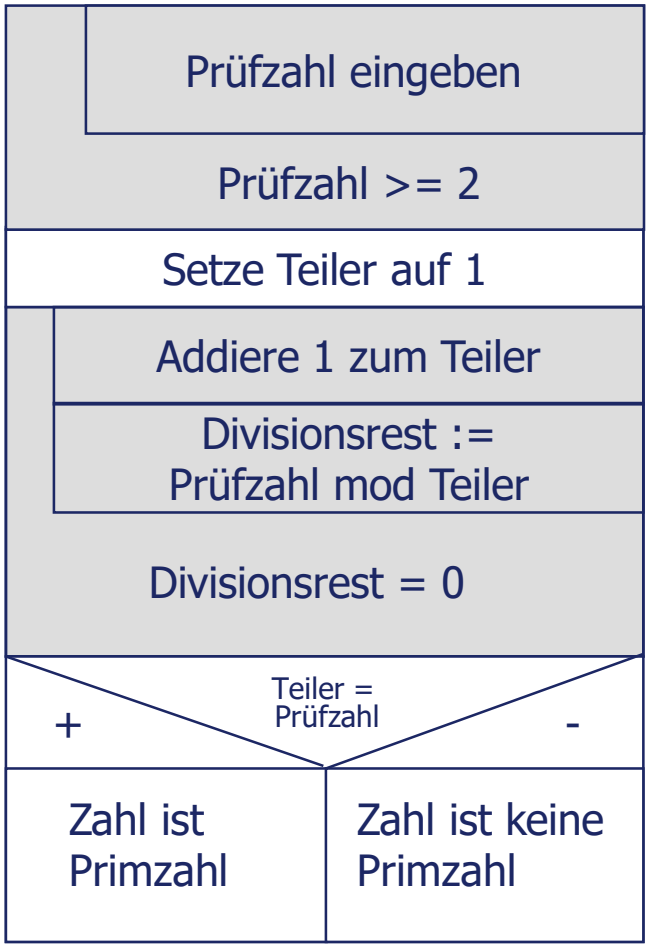
5 `repeat`-Anweisung

- Ausführung
  - Zunächst wird der Schleifenrumpf ausgeführt
  - Dann wird die boolesche Ausdruck nach dem Schlüsselwort `until` (Schleifenbedingung) ausgewertet
    - Ergebnis der Auswertung `False`:
      Erneute Ausführung der Anweisungen im Schleifenrumpf
    - Ergebnis der Auswertung `True`:
      Beenden der Ausführung der Repeat-Anweisung

  - Schleifenbedingung dient als *Abbruchbedingung*

5 `repeat`-Anweisung

- Beispiel: Test, ob eine Zahl (Prüfzahl) eine Primzahl ist



5 repeat-Anweisung

Beispiel: Test auf Primzahl als Programmtext

```
program Prim;
var
  Teiler, Pruefzahl, Rest : Word;
{$R+}
begin
  Repeat
    Write ('Eingabe der zu prüfende Zahl (>= 2): ');
    readln (Pruefzahl);
  until Pruefzahl >= 2;

  Teiler := 1;
  Repeat
      Teiler := Teiler + 1;
      Rest := Pruefzahl mod Teiler;
  until Rest = 0;
  if Teiler = Pruefzahl then
      writeln ('Zahl ist eine Primzahl!')
  else
      writeln ('Zahl ist keine Primzahl!');
  readln;
end.
```

5 `while`-Anweisung

- while-Anweisungen dienen zur Realisierung der Iteration mit Eintrittsbedingung
- Syntaktischer Aufbau:

Achtung, syntaktische Feinheiten!

- Mehrere Anweisungen im Schleifenrumpf müssen durch **Verbundanweisung** zusammengefasst werden (Klammerung durch `begin` und `end`)
- Unterschied zur Repeat-Anweisung!!

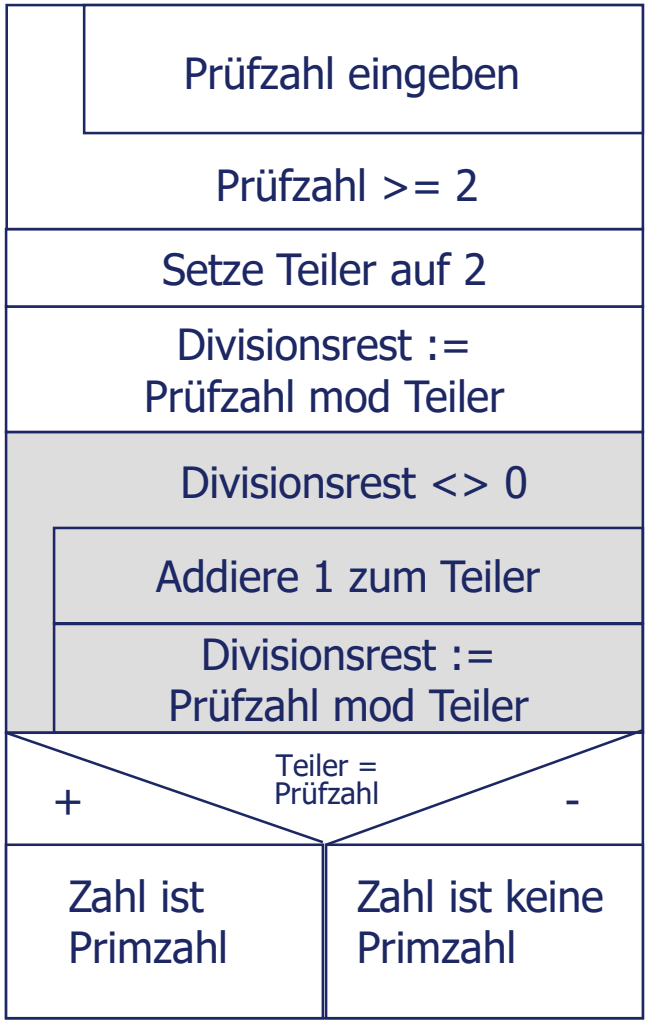
5 `while`-Anweisung

- Ausführung
  - Zunächst wird die Schleifenbedingung ausgewertet
  - Wenn das Ergebnis `True` ist:
    - Dann wird der Schleifenrumpf ausgeführt
    - Nach Ausführung des Schleifenrumpfes wird erneut die Schleifenbedingung ausgewertet
  - Wenn das Ergebnis `False` ist:
    - Dann wird die While-Anweisung ohne (erneute) Ausführung des Schleifenrumpfes verlassen

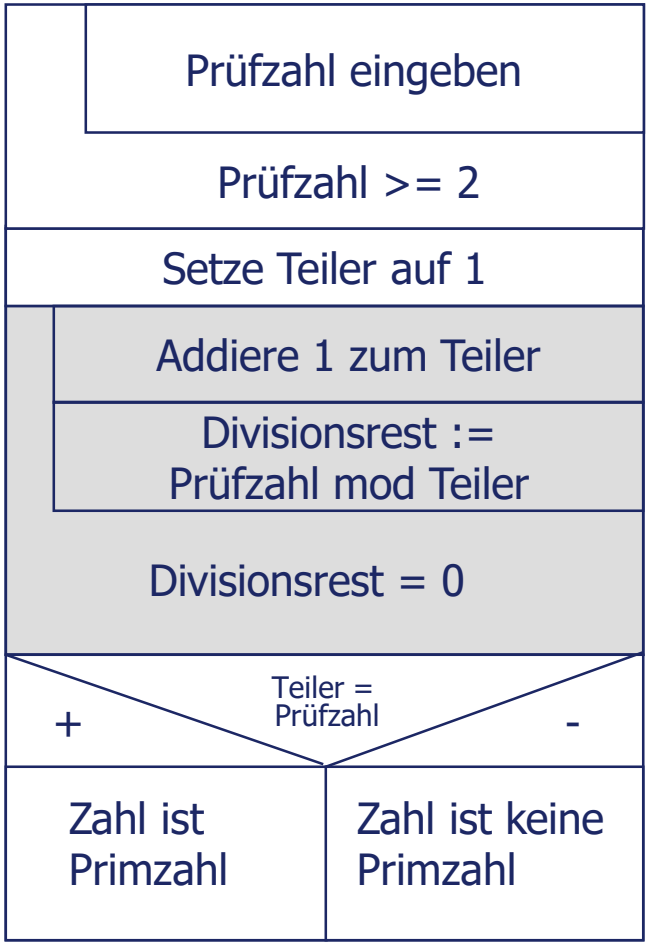
  - Schleifenbedingung dient als *Eintrittsbedingung*

5 `while`-Anweisung

Beispiel: Test auf Primzahl



Zur Erinnerung: Mit Abbruchbedingung



5 while-Anweisung

Beispiel: Test auf Primzahl als Programmtext

```
program Prim2;
var
  Teiler, Pruefzahl, Rest : Word;
{$R+}
begin
  Repeat
    write ('Eingabe der zu prüfende Zahl (>= 2): ');
    readln (Pruefzahl);
  until Pruefzahl >= 2;
  Teiler := 2;
  Rest := Pruefzahl mod Teiler;
  while Rest <> 0 do
  begin
      Teiler := Teiler + 1;
      Rest := Pruefzahl mod Teiler;
  end;
  if Teiler = Pruefzahl then
      writeln ('Zahl ist eine Primzahl!')
  else
      writeln ('Zahl ist keine Primzahl!');
  readln;
end.
```

5 Bezüge `while`-Anweisung und `repeat`-Anweisung

- Wechselseitige Realisierung der Wirkungsweise der Anweisungen möglich (insbes. Negation der Bedingung erforderlich):

```
while B do S;
```

```
if B then
   repeat
      S
   until not B;
```

```
repeat
   S
until B;
```

```
S;
while not B do S;
```

Aber nur diese Realisierungen sinnvoll!

5 Übung - Zahleigenschaften

In dieser Übung werden Zahlen auf „interessante“ Eigenschaften untersucht. Lesen Sie dazu vom Benutzer eine ganze Zahl zwischen `136` und `13010` ein und weisen Sie Ihn auf diese Grenzen hin! Die eingegebene Zahl wird auf die folgenden Eigenschaften geprüft:

- Die Anzahl der Ziffern in einem zur Übersetzungszeit festgelegten Zahlensystem. Dividieren Sie die Zahl so häufig durch die festgelegte Zahlenbasis bis diese 0 ist. Die Anzahl der Schleifendurchläufe entspricht dann der Anzahl der Stellen.

Beispiel für die Zahl 125 im Dezimalsystem (10)

```
125 div 10 = 12
 12 div 10 = 1
  2 div 10 = 0
```

Beispiel für die Zahl 125 im Binärsystem (2)

```
125 div 2 = 64
 64 div 2 = 32
 32 div 2 = 16
 16 div 2 = 8
  8 div 2 = 4
  4 div 2 = 2
  2 div 2 = 1
  1 div 2 = 0
```

5 Übung - Zahleigenschaften

Die weiteren beiden Eigenschaften können Sie in einer Schleife berechnen, Sie benötigen die Operatoren `div` und `mod` und noch einige zusätzliche Variablen.

- Ist die Zahl eine Schnapszahl? Schnapszahlen bestehen nur aus Wiederholungen einer einzigen Ziffer.
- Die Quersumme der Zahl, also die Addition aller Ziffern.

Ein Durchlauf des Programms könnte aussehen wie folgt:

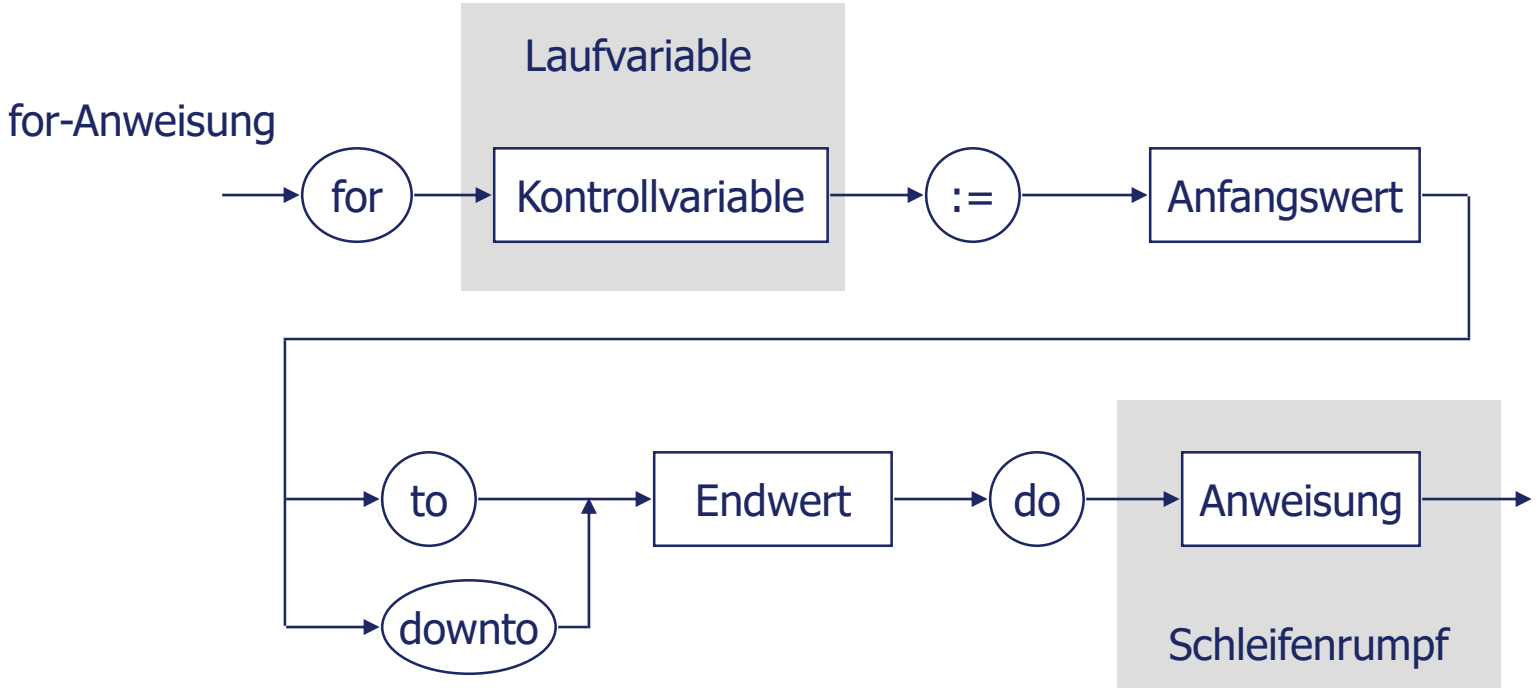
```
Ihre Zahl zwischen 136 und 13010? 135
Ihre Zahl zwischen 136 und 13010? 1456
4 Stellen zur Basis 10
Schnapszahl? FALSE
Quersumme: 16
```

5 for-Anweisung

- Spezielle Form der Wiederholungsanweisung für **Zählschleifen**
- Einschränkungen:
  - Schleifenkörper muss für *aufeinander folgende Werte eines ordinalen Datentyps* ausgeführt werden
  - Werte des ordinalen Datentyps, für die Schleifendurchläufe ausgeführt werden sollen, müssen bei Beginn der Schleife bekannt sein (nachträgliche Veränderungen nicht möglich)
- Trotz der Einschränkungen häufig verwendete Schleife

5 for-Anweisung

Syntaktischer Aufbau:



5 for-Anweisung

- Hinweise zum korrekten Aufbau von for-Anweisungen
  - Die Laufvariable muss deklariert werden (wie jede Variable)
  - Der Datentyp der Laufvariablen muss ein *ordinaler* Datentyp sein
  - Der Anfangswert und der Endwert (bzw. die entspr. Ausdrücke) müssen *zuweisungskompatibel* mit dem Datentyp der Laufvariablen sein
  - Sollen im Schleifenrumpf mehrere Anweisungen ausgeführt werden, muss eine *Verbundanweisung* verwendet werden

5 for-Anweisung

- Ausführung einer **for-*to***-Anweisung:
  - Der Anfangs- und der Endwert werden durch Auswertung der Ausdrücke ermittelt
  - Der Anfangswert wird der Laufvariablen zugewiesen
  - Wenn der Wert der Laufvariablen größer als der Endwert ist, wird die Ausführung der for-to-Anweisung beendet
  - Sonst werden die Anweisungen im Schleifenrumpf ausgeführt
  - Prüfung des Wertes der Laufvariablen:
    - wenn er gleich dem Endwert ist, wird die for-to-Anweisung beendet
    - wenn er kleiner als der Endwert ist, wird er auf den nächst größeren Wert des ordinalen Datentyps gesetzt und ein erneuter Durchlauf des Schleifenrumpfes begonnen

5 for-Anweisung

- Ausführung einer **for-*downto***-Anweisung:
  - Der Anfangs- und der Endwert werden durch Auswertung der Ausdrücke ermittelt
  - Der Anfangswert wird der Laufvariablen zugewiesen
  - Wenn der Wert der Laufvariablen kleiner als der Endwert ist, wird die Ausführung der for-downto-Anweisung beendet
  - Sonst werden die Anweisungen im Schleifenrumpf ausgeführt
  - Prüfung des Wertes der Laufvariablen
    - wenn er gleich dem Endwert ist, wird die for-downto-Anweisung beendet
    - wenn er größer als der Endwert ist, wird er auf den nächst kleineren Wert des ordinalen Datentyps gesetzt und ein erneuter Durchlauf des Schleifenrumpfes begonnen

5 for-Anweisung

Beispiel: Zinseszinsberechnung

```
program Zinsfor;

var
  Kapital, Zinssatz : Real;
  Jahre, Laufzeit : Byte;

begin
  write ('Bitte geben Sie das Anfangskapital ein: ');
  readln (Kapital);
  write ('Bitte geben Sie den Zinssatz ein: ');
  readln (Zinssatz);
  write ('Bitte geben Sie die Laufzeit ein: ');
  readln (Laufzeit);
  for Jahre := 1 to Laufzeit do
  begin
     Kapital := Kapital * (1 + Zinssatz / 100);
     writeln ('Kapital nach ', Jahre, ' Jahr(en):');
     writeln (Kapital:10:2, ' EUR');
     writeln;
  end;
  readln;
end.
```

5 for-Anweisung

- Zusätzliche Hinweise
  - **Der Wert der Laufvariablen darf in dem Schleifenrumpf nicht verändert werden** (In einigen Pascal-Versionen Zuweisungen an die Laufvariable nicht compilierbar!)
  - Eine for-Schleife kann (bei Beachtung der obigen Regel) nie zu einer Endlosschleife werden
  - Veränderungen von Anfangs- oder Endwert, die im Schleifenrumpf stattfinden, haben keine Auswirkungen
  - Nach Verlassen der for-Anweisung ist der Wert der Laufvariablen undefiniert (nicht gleich dem Endwert!)

5 Übung – Zahleigenschaften mit `for`-

Diese Übung darf auf dem Code der vorigen Aufgabenstellung aufbauen:

- Fragen Sie den Benutzer wie bisher nach einer Zahl (`pruefzahl`) zwischen `513` und `456122`.
- Fragen Sie den Benutzer dann nach zwei Zahlen (`pruefStart` und `pruefEnde`) zwischen `2` und `10`. Dabei muss die zweite eingegebene Zahl größer sein als die erste.
- Statt nur die Anzahl der Stellen für ein bestimmtes Zahlensystem auszugeben, sollen nun die Längen für alle Zahlenbasen von `pruefStart` bis `pruefEnde` ausgegeben werden.

Beispielausgabe:

```
Ihre Zahl zwischen 513 und 456122? 3
Ihre Zahl zwischen 513 und 456122? 514
Untere Grenze? 9
Obere Grenze? 10
3 Stellen zur Basis 9
3 Stellen zur Basis 10
```